8086 Addressing Modes

Dept. of Computer Science and Engineering
BRAC University
CSE 341 Team

Lecture References:

- **Book:**
  - *Microprocessors and Interfacing: Programming and Hardware, Chapter # 2,* **Author:** Douglas V. Hall
  - *The 8086/8088 Family: Design, Programming, And Interfacing, Chapter # 2,* **Author:** John Uffenbeck.

Addressing Mode and Categories

- The different ways in which a microprocessor can access data are referred to as its addressing modes.
- Addressing modes of 8086 Microprocessor are categorized as:
  - *Addressing Data*
  - *Addressing Program codes in memory*
  - *Addressing Stack in memory*
  - *Addressing I/O*
  - *Implied addressing*

Things to know…

- Instruction format

opcode	Operand(s)

- Instructions can have 1, 2 or no operands
  - **INC AX ;** 1 operand
  - **ADD CX, DX ;** 2 operands → CX = CX + DX
    Destination source
  - **HLT ;** no operand

- Instruction cannot have:
  - **SUB [DI], [1234h] ;** memory locations as both operands
  - **MOV 2345, 5425 ;** immediate data as both operands
  - **MOV 1234, AX ;** immediate data as destination operand

1. Addressing Data

I. Immediate addressing

II. Direct addressing

III. Register [direct] addressing

IV. Register indirect addressing

V. Base-plus-index addressing

VI. Register relative addressing

VII. Base-relative-plus-index addressing

1. Addressing Data

I. Immediate addressing

- Data is immediately given in the instruction

MOV BL, 11b

By default if no number system is given, we will consider it as decimal,d.

II. Direct addressing

- Data address is directly given in the instruction

MOV BX, [437AH]

1. Addressing Data

III. Register direct addressing

- Data is in a register (here BX register contains the data)

MOV AX, BX

IV. Register indirect addressing

- Register supplies the address of the required data

MOV CX, [BX]

MOV [SI], AX

1. Addressing Data

BRAC UNIVERSITY
Inspiring Excellence

v. Base-plus-index addressing

- Base register is either BX or BP
- Index register is either DI or SI

MOV DX, [BX+DI]

v. Register relative addressing

- Register can be a base (BX, BP) or an index register (DI, SI)
- Mainly suitable to address array data

MOV AX, [BX+1000]

1. Addressing Data

VII. Base-relative-plus-index addressing

- Suitable for array addressing

MOV AX, [BX+DI+10]

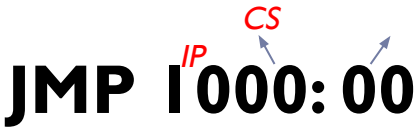
MOV [BX+DI+10],AX

2. Addressing Program Codes in Memory

- Used with JMP and CALL instructions
- 3 distinct forms:
  I. Direct
  II. Indirect
  III. Relative

2. Addressing Program Codes in Memory

- Address is directly given in the instruction



JMP doagain ; doagain is a **label** in code

CALL 1000:0000

CALL doagain ; doagain is a **procedure** in code

- Often known as *far* jump or *far* call

2. Addressing Program Codes in Memory

- Address can be obtained from
  - **a)** any GP registers (AX,BX,CX,DX,SP,BP,DI,SI)

    JMP AX *IP = AX ; then CS : IP*
  - **b)** any relative registers ([BP],[BX],[DI],[SI])

    JMP [BX] *IP = what is inside the physical address of DS : BX ; then CS : IP*
  - **c)** any relative register with displacement

    JMP [BX + 100h] *IP = what is inside the physical address of DS : BX +100h ; then CS : IP*

Finding Location 8086 jump into

Step 1: Finding location where IP is located.

This will follow the normal rule as physical addressing

Step 2: Take the value from the location as IP value of 16 bit.

Step 3: (CSx10) + IP

Address	31234h	31235h	12000h	12001h	30600h	30601h
Data	12h	34h	10h	20h	11h	21h

A. Assume for an 8086, **DS = 1000h, CS = 3000h, SS = 8A40h, BX = 2000h, BP = 1234h, SI = 0020h, DI = 030Fh.** We also execute the **JMP [BX]** instruction. **Now, deduce** the **physical address** of the memory location 8086 will jump to. **[4]**

Location where IP is located = DSx10 + BX = 10000h + 2000h = 12000h

Value of IP = 2010h

Location where 8086 will jump to = CSx10 + IP = 30000 + 2010h = 32010h.

Address	31234h	31235h	30600h	30601h	40600h	40601h
Data	12h	34h	11h	21h	30h	40h

8

A. Assume for an 8086, **DS = 4000h, CS = 3000h, SS = 8A40h, BX = 2000h, BP = 1234h, SI = 600h, DI = 030Fh.** We also execute the **JMP [SI]** instruction. **Now, deduce** the **physical address** of the memory location 8086 will jump to. **[4]**

Answer the questions based on the given table and data

Physical Address	8 bit hex data
14001h	78h
14000h	56h
13001h	34h
13000h	12h
03000h	ABh

Given DS = **1000h**, CS = **3000h**, SS = **8A40h**, SI = **2000h**, CX = **43AEh**, DI = **030Fh**. We also execute the **JMP [SI + 1000h]** instruction. Now, answer the questions based on the given table and data:

3. Addressing Stack in Memory

- **PUSH** and **POP** instructions are used to move data to and from stack (in particular from stack segment).

  PUSH AX → Push data from AX to stack's top

  POP CX → Pops data from stack's top to CX

- **CALL** also uses the stack to hold the return address for procedure.

  CALL SUM ; SUM is a procedure name

Push Pop in Stack

1. PUSH BX or MOV [SS+SP] , BX

Pushes / transfers value from BX register to Stack top

2. POP DX or MOV DX, [SS+SP]

Pops / transfers value from Stack's top to the DX register.

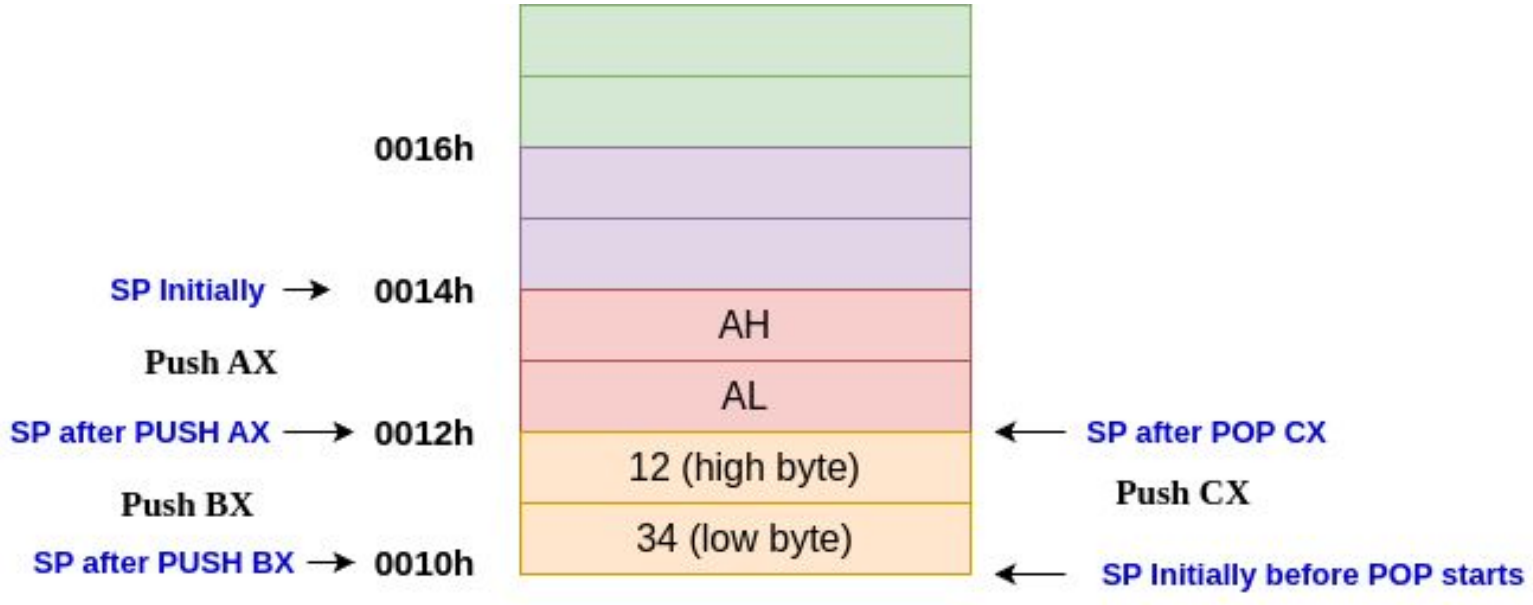
Stack

Push once → SP value is decreased by 2 hex , and 2 byte of data is pushed into the stack.

Pop Once → 2 bytes of data popped out of the stack, then SP value is increased by 2 hex

[SS + SP] → Top of the stack

Here the push operations are done first and then pop operation



BX register holds 1234h.
So after PUSH BX, high byte has SP=0011h and low byte has SP = 0010h

After POP CX,
CX holds 1234h

MOV AX, [SS+SP] or POP AX

1. **Given DS = 1000h, CS = 3000h, SS = 2000h, SI = 1005h, IP = 1234h, SP = 2342h.**

Address	Data
22345	12h
22344	23h
22343	34h
22342	45h
22341	56h

A. **Deduce** the instruction that loads data from the top of the stack into the AX register. [1 mark]

A. SP points to top of the stack. So

Location the top of the stack points to→ (SSx10) + SP

= 20000 + 2342

= 22342h

Data in AX = 3445h

4. Addressing Input and Output Port

- IN and OUT instructions are used to address I/O ports
- Could be *direct addressing*

  IN AL, 05h ; Here 05h is a input port number

- or *indirect addressing*

  OUT DX, AL ; DX contains the address of I/O port

- Only DX register can be used to point a I/O port

5. Implied Addressing

- No explicit address is given with the instruction
- implied within the instruction itself
- Examples:

```
CLC  ; clear carry flag
HLT  ; halts the program
RET  ; return to DOS
```

8086 Machine Codes

Dept. of Computer Science and Engineering
BRAC University
CSE 341 Team

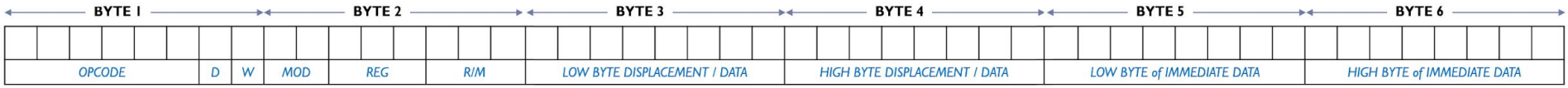
Instruction template

- For 8085: Just look up the hex code for each instruction.
- For 8086 it is not simple.
- E.g 32 ways to specify the source in **`MOV CX, source`**.
- MOV CX, **source**

a 16-bit register (8 in number)

a memory location (24 possible memory addressing modes)

- Each of these 32 instructions require different binary code.
- Impractical to list them all in a table.
- Instruction templates help code the instruction properly.



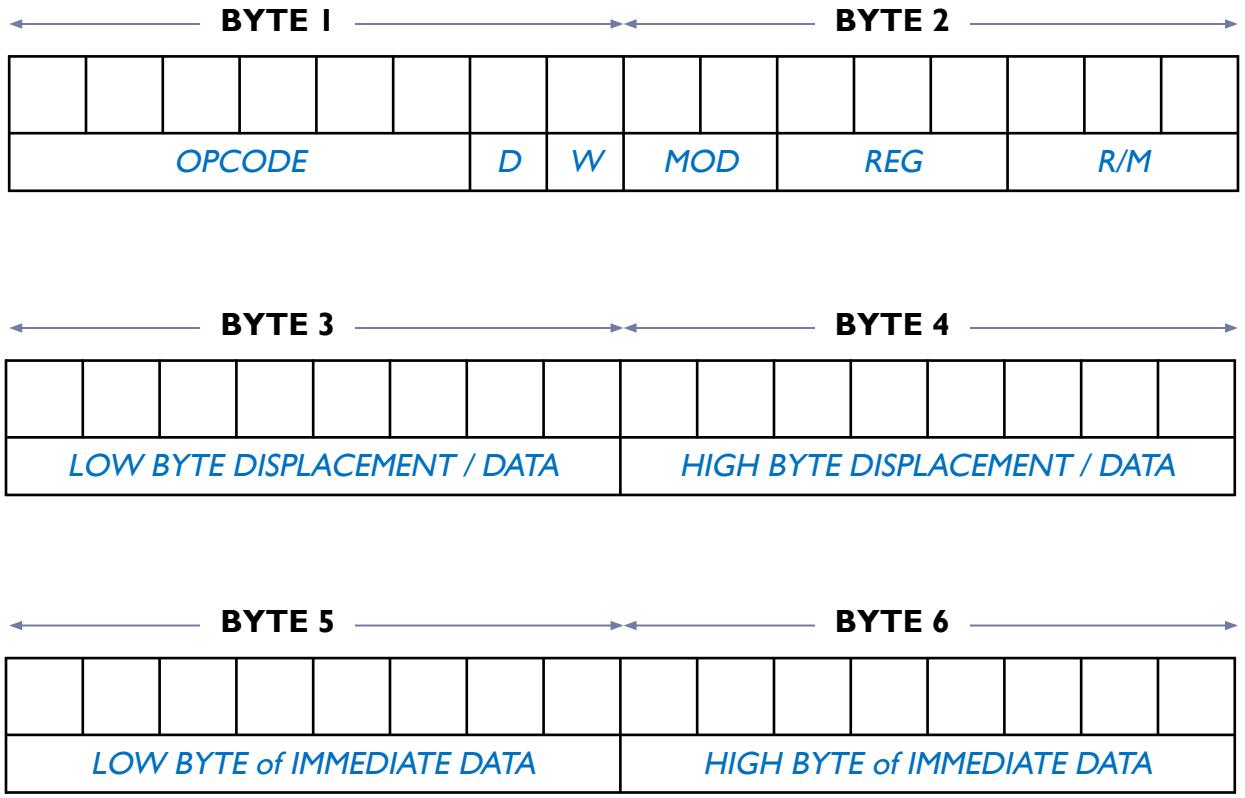
Testing the validity before machine code generation

BRAC UNIVERSITY
Inspiring Excellence

- There will be at least one register.
- There can NOT be a immediate value in the destination.
- There will be a specific offset for a particular segment.

IF the instruction is INVALID, then the machine code would NOT be generated.

Instruction template (6 bytes)



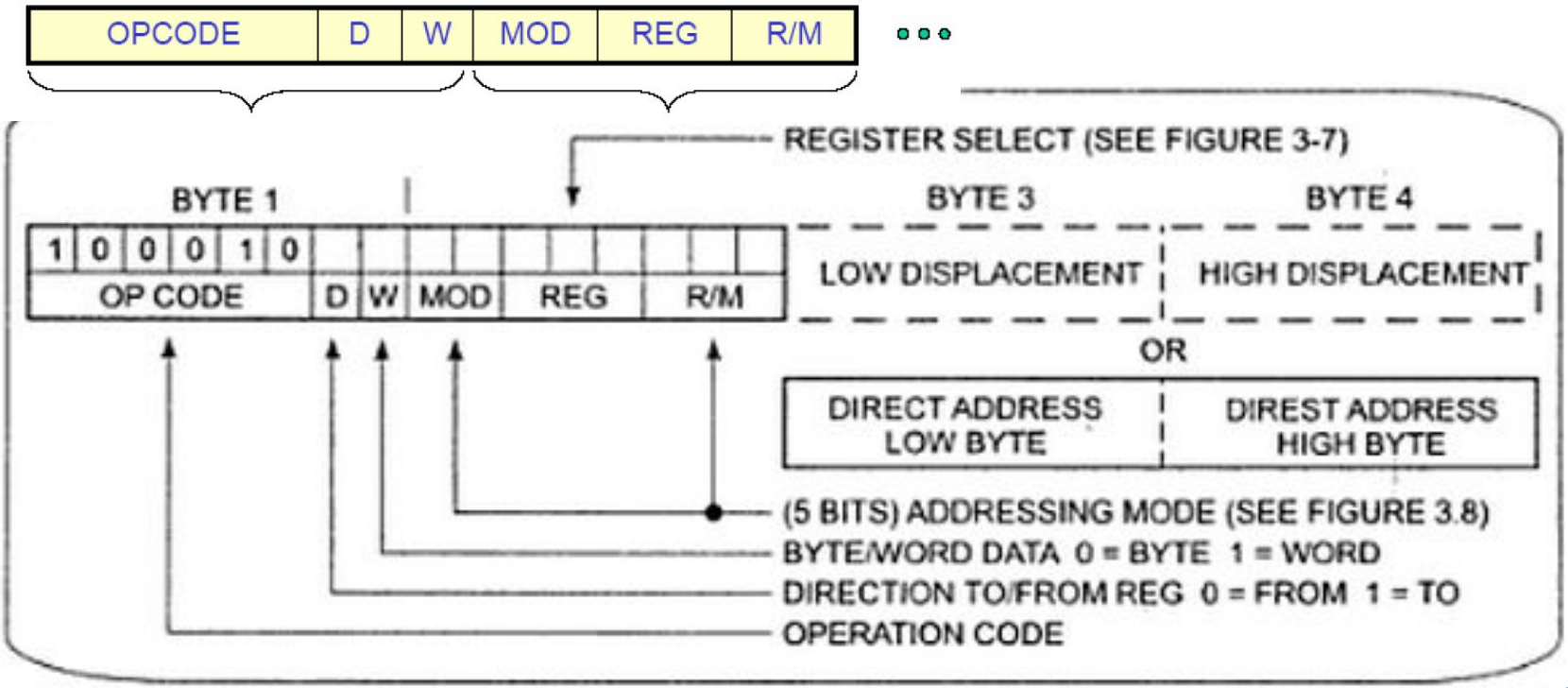
An instruction after conversion can have 1 to 6 bytes long of machine code

Constructing Machine Codes for 8086

- Each instruction in 8086 is associated with the binary code.
- You need to locate the codes appropriately.
- Most of the time this work will be done by assembler
- The things needed to keep in mind is:
  - Instruction templates and coding formats
  - MOD and R/M Bit patterns for particular instruction

MOV Instruction Coding

- MOV data from a register to a register/to a memory location or from a memory location to a register. (**Operation Code of MOV: 100010**)



MOD and R/M Field

- 2-bit Mode (MOD) and 3-bit Register/Memory (R/M) fields specify the other operand.
- Also specify the addressing mode.

MOD / RM	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BXI + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] +d16	AH	SP
101	[DI]	[DI] + d8	[DI] +d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] +d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

MOD and R/M Field

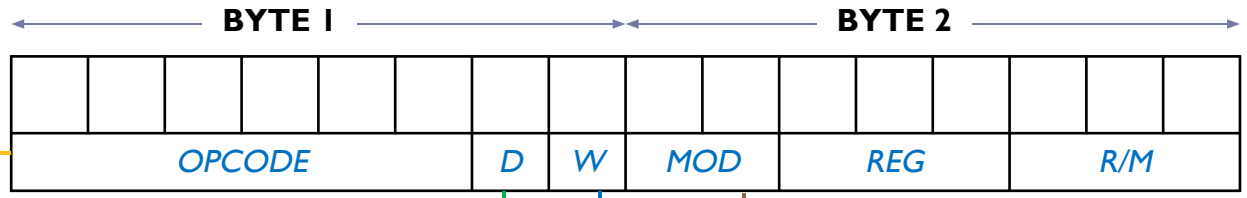
- If the other operand in the instruction is also one of the eight register then put in 11 for MOD bits in the instruction code.
- If the other operand is memory location, there are 24 ways of specifying how the execution unit should compute the effective address of the operand in the main memory.
- If the effective address specified in the instruction contains displacement less than 256 along with the reference to the contents of the register then put in 01 as the MOD bits.
- If the expression for the effective address contains a displacement which is too large to fit in 8 bits then out in 10 in MOD bits.

REG Field

- REG field is used to identify the register of the one operand

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Instruction template



6 bits of MOV, ADD etc

D - direction
If **D=0**, then direction is from a register (source)
If **D=1**, then direction is to a register (destination)

W - word
If **W=0**, then only a byte is being transferred (8 bits)
If **W=1**, them a whole word is being transferred (16 bits)

MODE	OPERAND NATURE	
00	Memory with no displacement	MOV AX, [BX]
01	Memory with 8-bit displacement	MOV AX, [BX + 12h]
10	Memory with 16-bit displacement	MOV AX, [BX + 1234h]
11	Both are registers	MOV AX, BX

- 34h here is an 8-bit displacement
- [BX+34h] is a memory/offset address

MOV [BX + 34h],AL

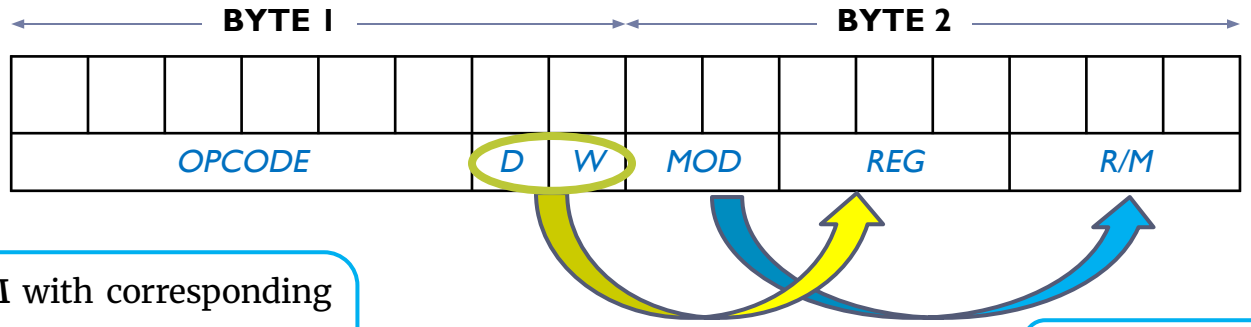
MOV AX, 1234 h

- 1234h here is a 16-bit immediate data value

Instruction	D	W	MOD
MOV BL , [1234H]	1	0	00
MOV [DX], BX	0	1	00
MOV [BX + SI + 1000H], AX	0	1	10
MOV AL, [DI + 34H]	1	0	01
MOV CX, DX	0/1	1	11
MOV [2344H], [1234H]	Not Valid		
MOV 2344H, CX	Not Valid		

$$[DI + 34h] = [DI]34h = 34h[DI]$$

Instruction template



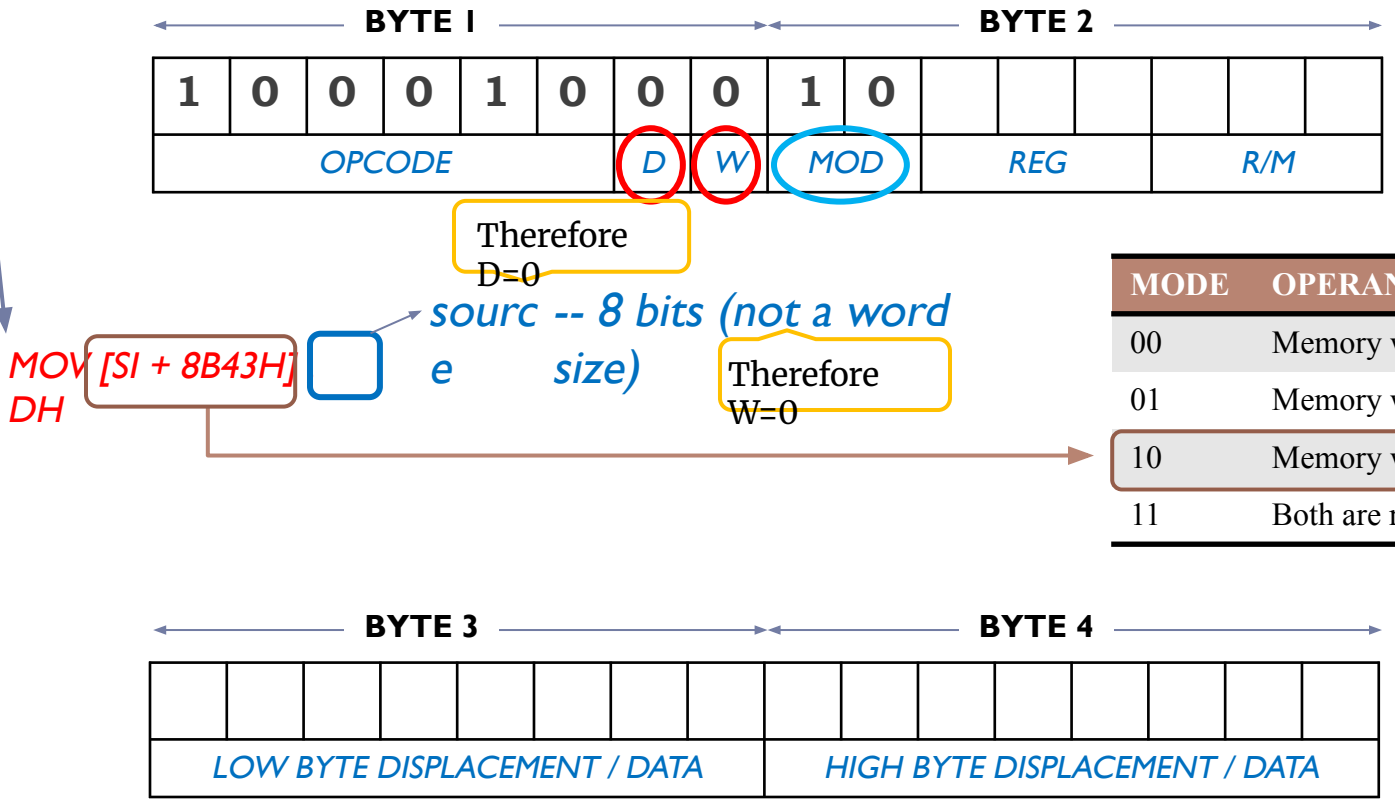
- Value for **R/M** with corresponding MOD value
- Value for REG with corresponding **W** value and the register considered in D

Check column that matches with MOD value

RM \ MOD	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BXI + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] +d16	AH	SP
101	[DI]	[DI] + d8	[DI] +d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] +d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

Example 1

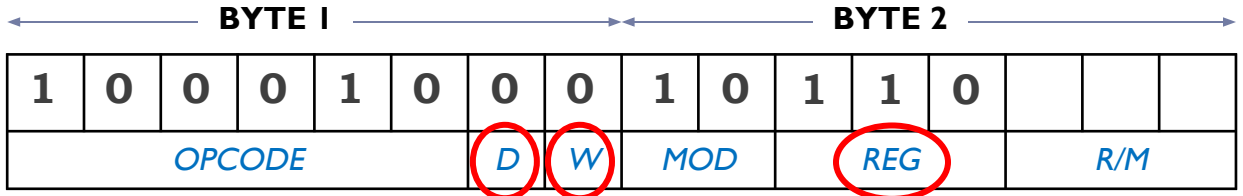
- **MOV 8B43H [SI], DH:** Copy a byte from DH to memory with 16 bit displacement given the opcode for MOV=100010



MODE	OPERAND NATURE
00	Memory with no displacement
01	Memory with 8-bit displacement
10	Memory with 16-bit displacement
11	Both are registers

Example 1

- **MOV 8B43H [SI], DH:** Copy a byte from DH to memory with 16 bit displacement given the opcode for MOV=100010

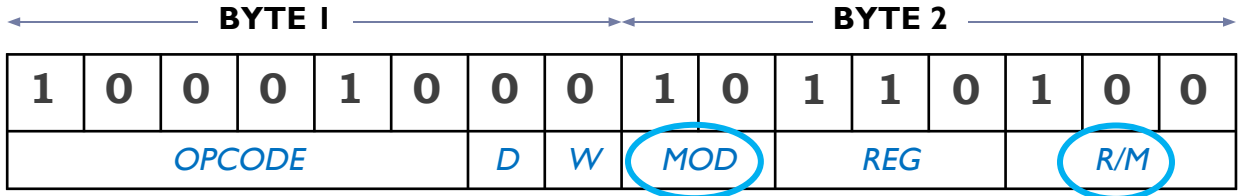


MOV [SI + 8B43H] , DH

MOD / RM	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BX] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] + d16	AH	SP
101	[DI]	[DI] + d8	[DI] + d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

Example 1

- **MOV 8B43H [SI], DH:** Copy a byte from DH to memory with 16 bit displacement given the opcode for MOV=100010

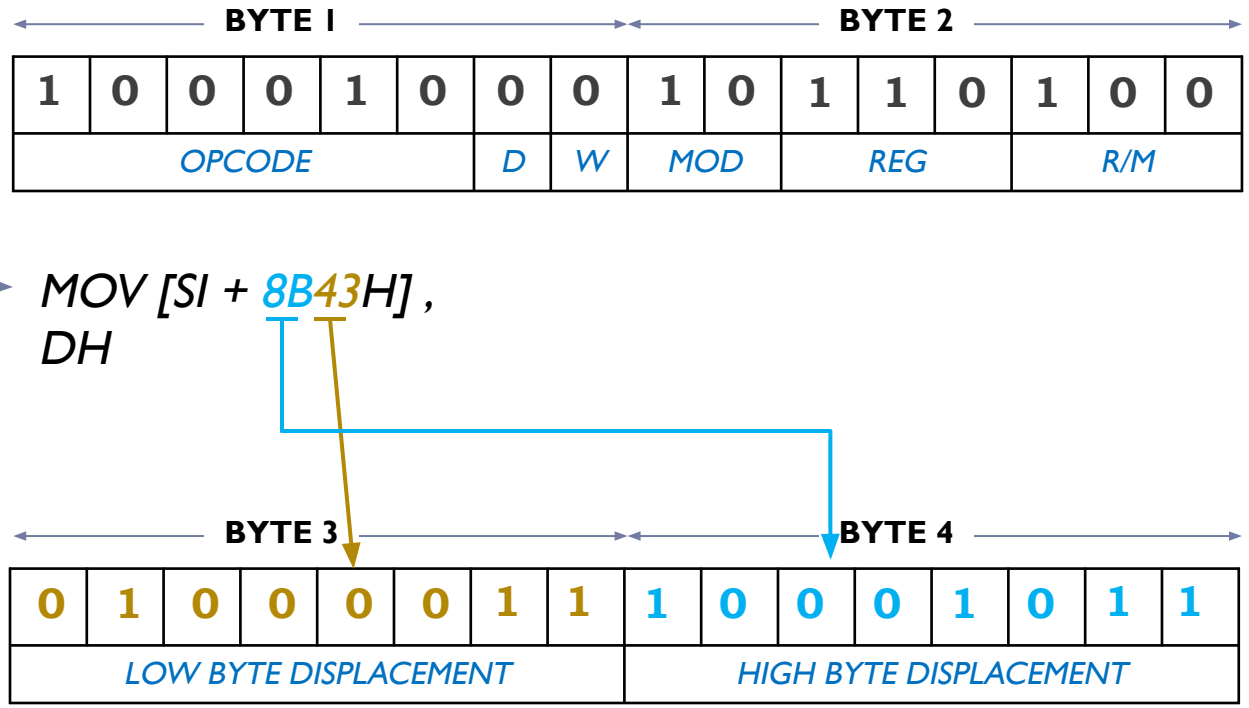


MOV [SI + 8B43H] , DH

RM \ MOD	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BX] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] + d16	AH	SP
101	[DI]	[DI] + d8	[DI] + d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

Example 1

- **MOV 8B43H [SI], DH:** Copy a byte from DH to memory with 16 bit displacement given the opcode for MOV=100010



Machine Code: *1000 1000 1011 0100 0100 0011 1000 1011*$_2$ *or 88 B4 43 8B*$_{16}$

Example 2

- **MOV [1234H], DL**

MOD / RM	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BXI + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] +d16	AH	SP
101	[DI]	[DI] + d8	[DI] +d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] +d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

OPCODE						D	W
1	0	0	0	1	0	0	0

MOD		REG			R/M		
0	0	0	1	0	1	1	0

Low 8 Bit / Low 1 Byte / Low 2 Hex Digit							
0	0	1	1	0	1	0	0

High 8 Bit / High 1 Byte / High 2 Hex Digit							
0	0	0	1	0	0	1	0

Example 3

BRAC UNIVERSITY
Inspiring Excellence

MOV DH, [BP+SI+7Dh]

MOD / RM	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BXI + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] +d16	AH	SP
101	[DI]	[DI] + d8	[DI] +d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] +d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

OPCODE						D	W
1	0	0	0	1	0	1	0

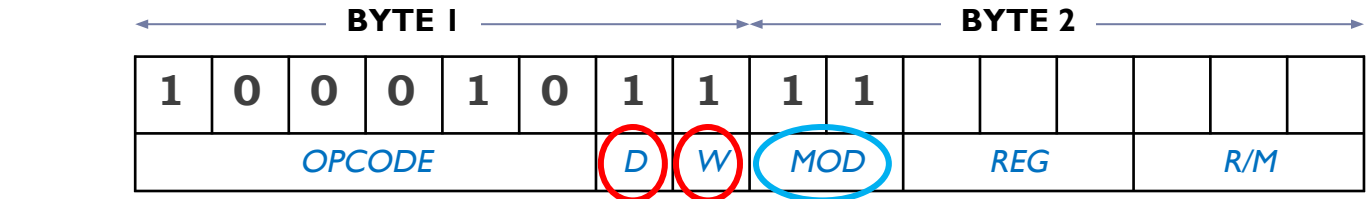
MOD		REG			R/M		
0	1	1	1	0	0	1	0

Low 8 Bit / Low 1 Byte / Low 2 Hex Digit							
0	1	1	1	1	1	0	1

There is no 4th Byte

Example 4

? **MOV AX, BX:** given the opcode for MOV=100010



AX considered which is a destination operand, therefore D = 0

destinatio -- 16 bits (a word size)

AX is 16-bit long, therefore W = 1

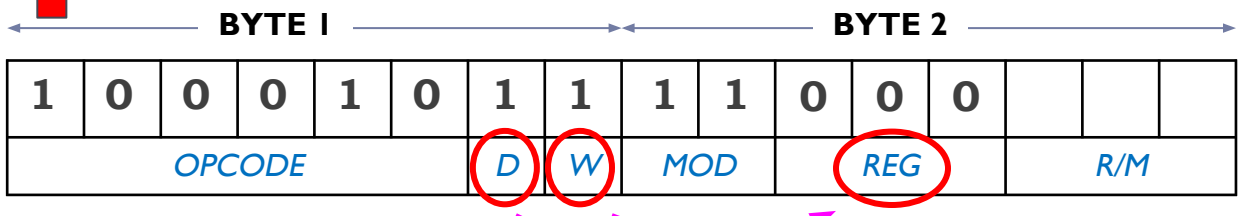
MOV AX, BX

MODE	OPERAND NATURE
00	Memory with no displacement
01	Memory with 8-bit displacement
10	Memory with 16-bit displacement
11	Both are registers

Example 4

- **MOV AX, BX:** given the opcode for MOV=100010

Considering AX to be our main register

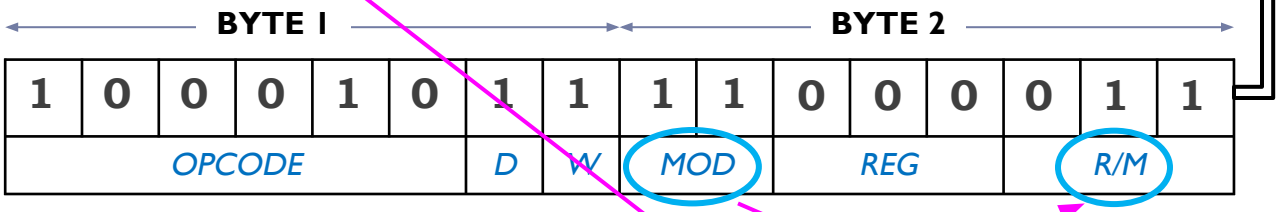


MOD / RM	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BXI + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] +d16	AH	SP
101	[DI]	[DI] + d8	[DI] +d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] +d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

Example 4

Machine Code: *1000 1011 1100 0011*$_2$ *or 8B C3*$_{16}$

- **MOV AX, BX:** given the opcode for MOV=100010



MOD / RM	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BX] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] + d16	AH	SP
101	[DI]	[DI] + d8	[DI] + d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

Example 4 (Alternative)

- **MOV AX, BX**

Now Considering BX to be our main register

OPCODE						D	W
1	0	0	0	1	0	0	1

Here, since BX is our MAIN register and it at source so D = 0

MOD		REG			R/M		
1	1	0	1	1	0	0	0

Here for REG, we are considering BX

Here for R/M, we are considering AX

Example 5

- **MOV 1234H, DL**

MOD / RM	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BXI + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] +d16	AH	SP
101	[DI]	[DI] + d8	[DI] +d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] +d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

Not valid. No machine code.

Machine Code Size

BRAC UNIVERSITY
Inspiring Excellence

MOD = 00

1. MOV [1234H], DL → 4 bytes
2. MOV AX, [BX] → 2 bytes

MOD = 01

MOV DH, [BP + 7Dh] → 3 bytes

MOD = 10

MOV DH, [BP + 712Dh] → 4 bytes

BRAC UNIVERSITY
Inspiring Excellence

MOD = 11

MOV DH, AL → 2 bytes

- **89807812h to instruction**

- **8A167812h to instruction**

BRAC UNIVERSITY
Inspiring Excellence

Thus, the instruction will be MOV [BX+SI+1278h], AX

Find the address mode, mod value and byte length

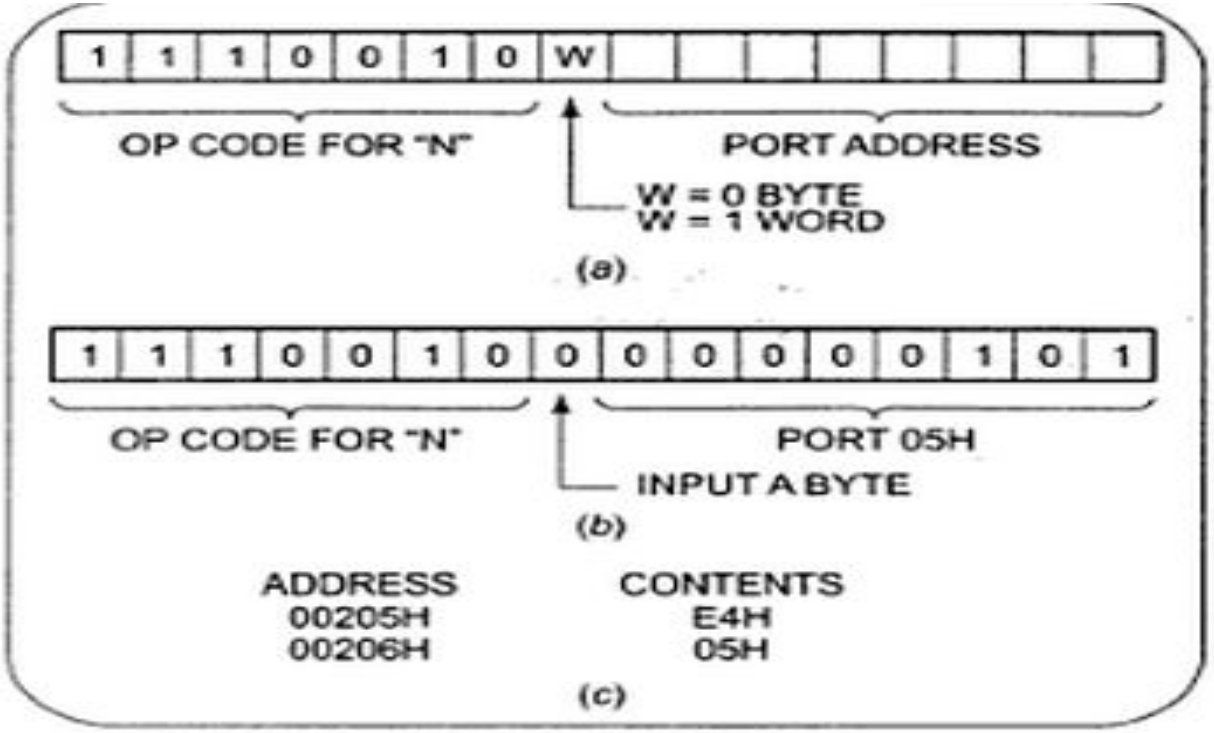
BRAC UNIVERSITY
Inspiring Excellence

Instruction	Addressing Mode	Mod	Byte Length
MOV AH, BL	Register Direct Addressing	11	2
MOV [SI], BL	Register Indirect Addressing	00	2
MOV CX, [BP + SI + 3452h]	Base relative plus index Addressing	10	4
MOV [AH], [BL]	Not Valid	-	-
MOV [BX + DI], AX	Base plus index Addressing	00	2
MOV [BX + 22h], AX	Base plus relative	01	3
MOV AX, [1422h]	Direct Addressing	00	4
MOV 1234h[SI], AX	Base plus relative	10	4

Instruction Template

- The Intel literature shows two different formats for coding 8086 instructions.
- Instruction templates helps you to code the instruction properly.
- **Example:**

  IN AL, 05H



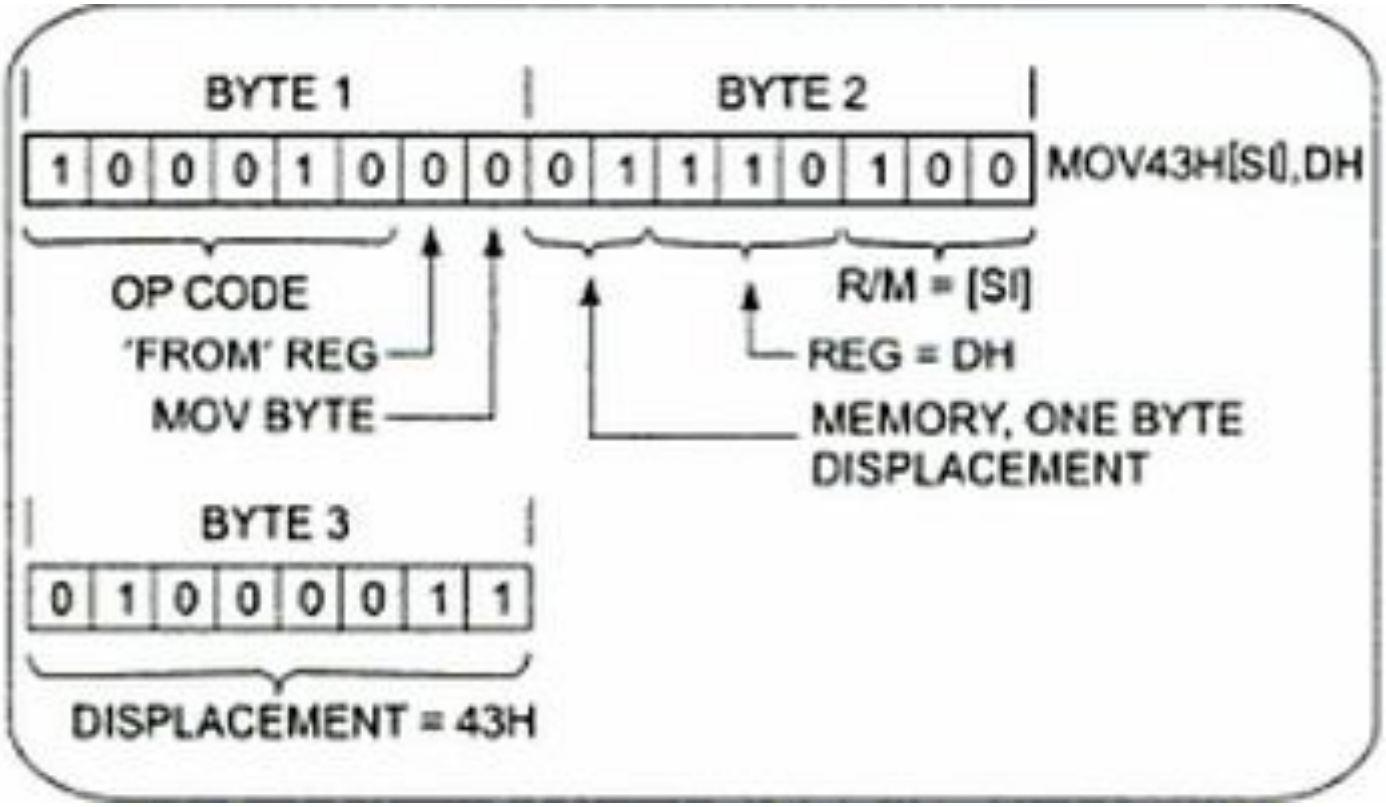
Example

- MOV BL,AL
- Opcode for MOV = 100010
- We'll encode AL so
  - D = 0 (AL source operand)
- W bit = 0 (8-bits)
- MOD = 11 (register mode)
- REG = 000 (code for AL)
- R/M = 011

OPCODE	D	W	MOD	REG	R/M
100010	0	0	11	000	011

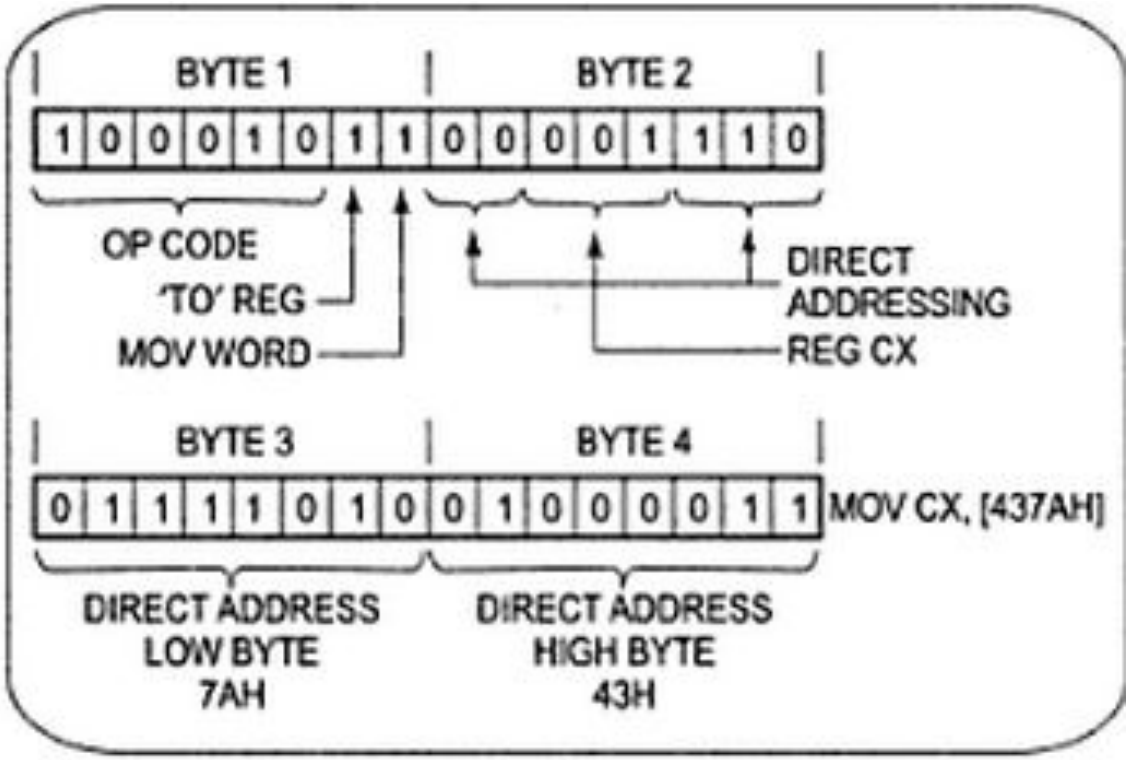
Example

- **MOV 43H [SI], DH:** Copy a byte from DH register to memory location.



Example 3

- **MOV CX, [437AH]:** Copy the contents of the two memory locations to the register CX.



BRAC UNIVERSITY
Inspiring Excellence

Thank You !!!